

CLAIMS

WE CLAIM:

1. A method for using a mirror code process to analyze a managed code process, the method comprising:

running the managed code process in a runtime environment, whereby the running managed code process writes to and reads from a first address space while running;

running the mirror code process, whereby the running mirror code process writes to a second address space while running, the second address space not overlapping the first address space; and

using cross-process memory access by the mirror code process to write at least a portion of the contents of the first address space into the second address space.

2. The method according to claim 1, further comprising the step of analyzing the portion of the contents of the first address space written into the second address space by the mirror code process, to identify flaws with respect to the managed code process.

3. The method according to claim 1, wherein using cross-process memory access by the mirror code process to write at least a portion of the contents of the first address space into the second address space comprises caching by the mirror code process at least a portion of the contents of the first address space.

4. The method according to claim 3, wherein caching by the mirror code process at least a portion of the contents of the first address space comprises constructing a cache that omits duplicate addresses from first address space overlapping address ranges.
5. The method according to claim 3, wherein caching by the mirror code process at least a portion of the contents of the first address space comprises constructing a cache that includes duplicate addresses from first address space overlapping address ranges.
6. The method according to claim 1, wherein the runtime environment and the mirror code process run on different first and second respective computing devices, and wherein using cross-process memory access by the mirror code process comprises retrieving information at the second computing device from the first computing device.
7. The method according to claim 6, wherein the first and second computing devices have CPUs of mutually distinct types..
8. A computer-readable medium having thereon computer-executable instructions for performing the method of claim 1.
9. A computer-readable medium having thereon computer-executable instructions for performing the method of claim 2.

10. A computer-readable medium having thereon computer-executable instructions for performing the method of claim 7.

11. A method of compiling a source code body of code into a mirror code body of code comprising:

analyzing the source code body to identify in-process pointers; and
replacing in the mirror code body at least a portion of the identified in-process pointers with cross-process pointers such that when the mirror code body is run in a first process the cross-process pointers are usable to read information from a memory of a second process.

12. A computer-readable medium having thereon computer-executable instructions for performing the method of claim 11.

13. The method according to claim 11, wherein replacing at least a portion of the identified in-process pointers with cross-process pointers further comprises placing in the mirror code body a retrieval routine associated with each such cross-process pointer, wherein the retrieval routine is operable to use the respective cross-process pointer to read from the memory of the second process.

14. A computer-readable medium having thereon computer-executable instructions for performing the method of claim 13.

15. The method according to claim 11, wherein replacing in the mirror code body at least a portion of the identified in-process pointers with cross-process pointers further comprises:

scanning a listing of global data addresses in the source code body of code;
determining each global address in the memory of the second process; and
encoding each such global address in the mirror code body as data.

16. The method according to claim 11, wherein replacing in the mirror code body at least a portion of the identified in-process pointers with cross-process pointers further comprises utilizing a vtable pointer to identify the size of a class to be mirrored by the mirror code body.

17. The method according to claim 11, further comprising additionally compiling the source code body of code into runtime code body, wherein in-process pointers in the source code remain as in-process pointers in the runtime code body.

18. The method according to claim 17, further comprising:

running the mirror code body in the first process, whereby the first process writes to and reads from a first memory space while running;
running the runtime code body in the second process via a runtime environment, whereby the second process writes to and reads from the memory of the second process while running; and

using cross-process memory access by the first process to write at least a portion of the contents of the memory of the second process into the first memory space.

19. A system for compiling a source code body of code into a mirror code body of intermediate language code comprising:

means for analyzing the source code body to identify in-process pointers; and
means for replacing in the mirror code body at least a portion of the identified in-process pointers with cross-process pointers such that when the mirror code body is run in a first process the cross-process pointers are usable to read information from a memory of a second process.